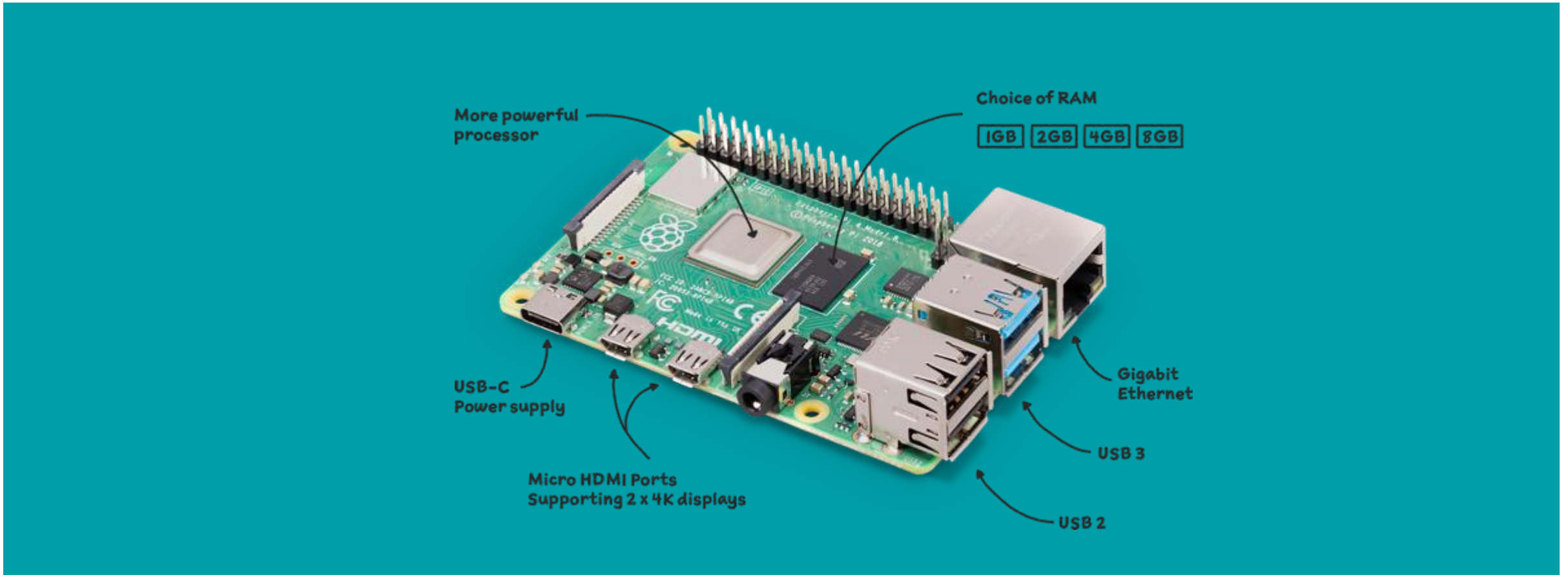
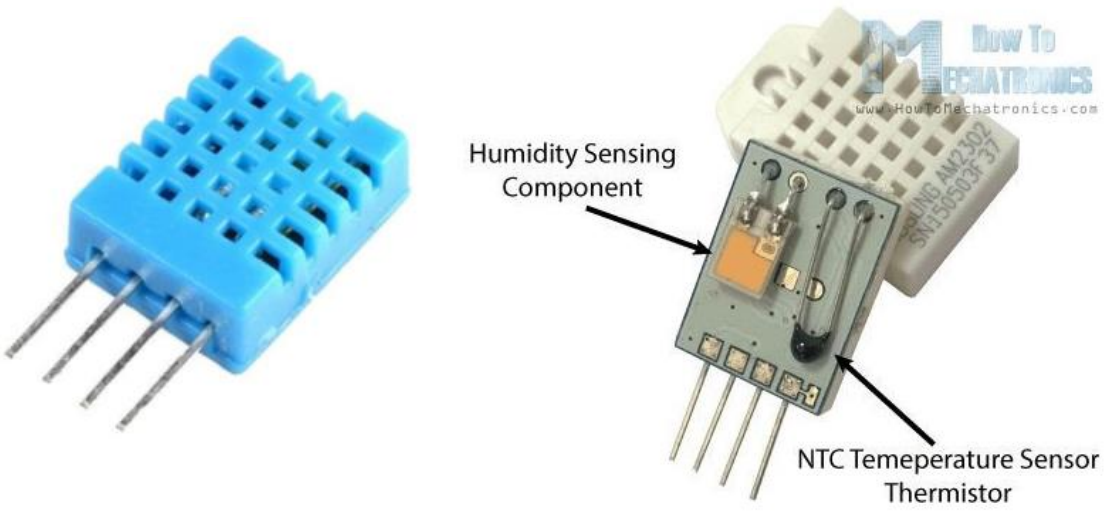


3. 센서활용

이영주
young.kopo@gmail.com

한국폴리텍대학

온습도센서 제어



3. Typical Application (Figure 1)

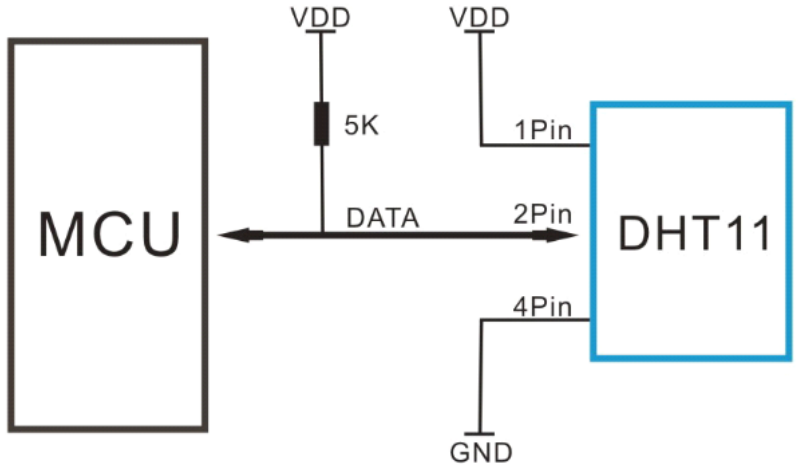


Figure 1 Typical Application

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

온도센서

온도센서:

서미스터(Thermistor: thermally sensitive resistor)라 하는 반도체의 저항이 온도에 따라 변하는 특성을 이용

서미스터의 저항온도계수(TCR: Temperature Coefficient of Resistance)

- 일정하지 않고 온도에 따라 달라지는 특성

정온도 계수(PTC:positive temperature coefficient)형:

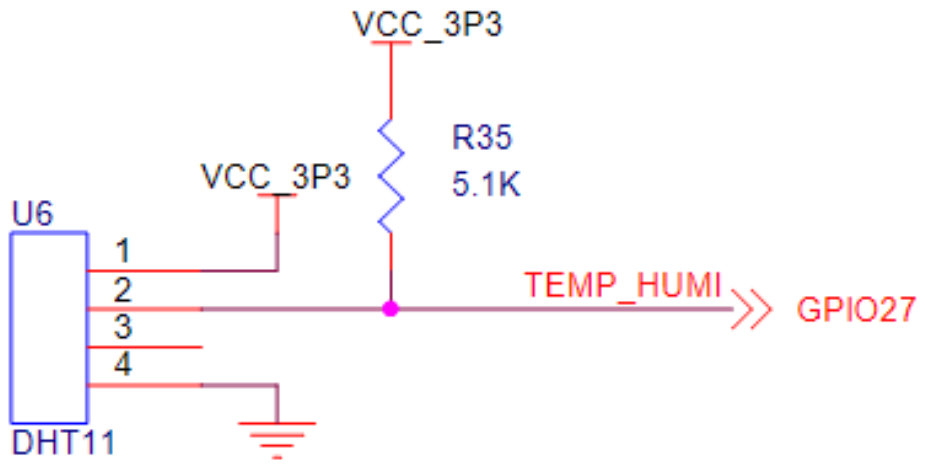
- 온도가 증가함에 따라 저항이 증가 하는 타입

부온도 계수(NTC:negative temperature coefficient)형:

- 온도가 증가함에 따라 저항은 감소하는 타입

DHT11은 NTC형이 사용됨

연결회로 및 통신방식



BCM: 27
WiringPi: 2

5. Communication Process: Serial Interface (Single-Wire Two-Way)

Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms.

Data consists of decimal and integral parts. A complete data transmission is **40bit**, and the sensor sends **higher data bit** first.

Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

통신방식

5.1 Overall Communication Process (Figure 2, below)

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low-power-consumption mode until it receives a start signal from MCU again.

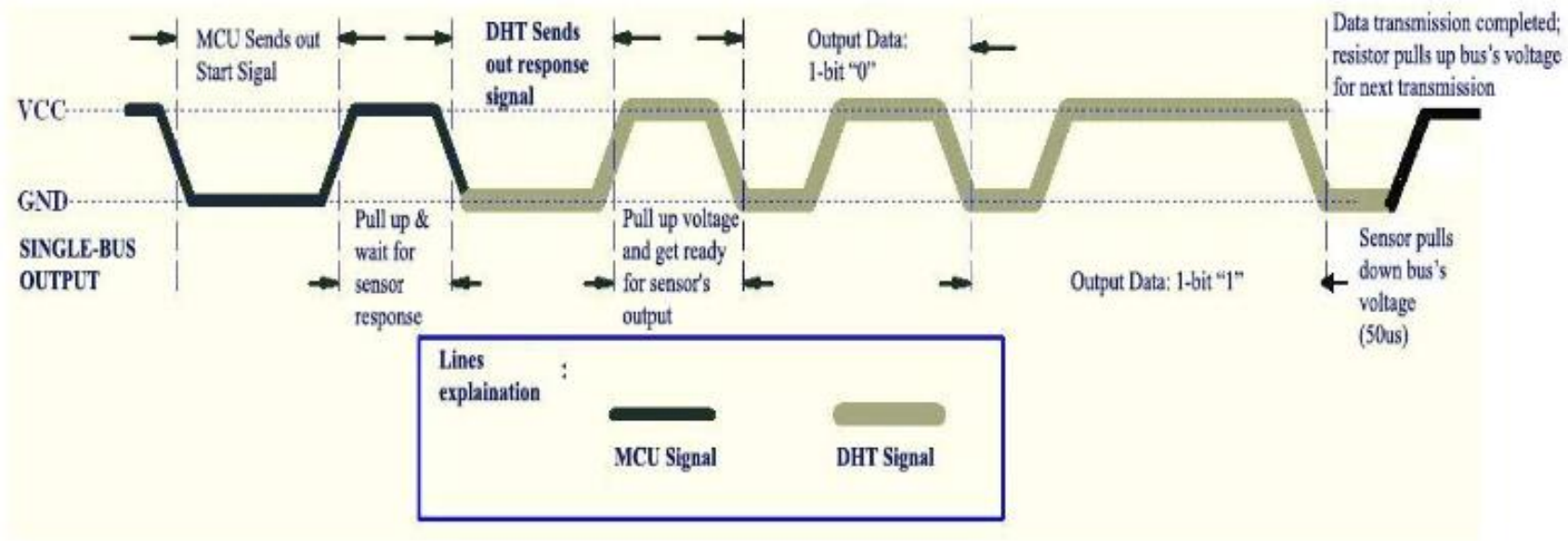


Figure 2 Overall Communication Process

통신방식

5.2 MCU Sends out Start Signal to DHT (Figure 3, below)

Data Single-bus free status is at high voltage level. When the communication between MCU and DHT11 begins, the programme of MCU will set Data Single-bus voltage level from high to low and this process must take at least 18ms to ensure DHT's detection of MCU's signal, then MCU will pull up voltage and wait 20-40us for DHT's response.

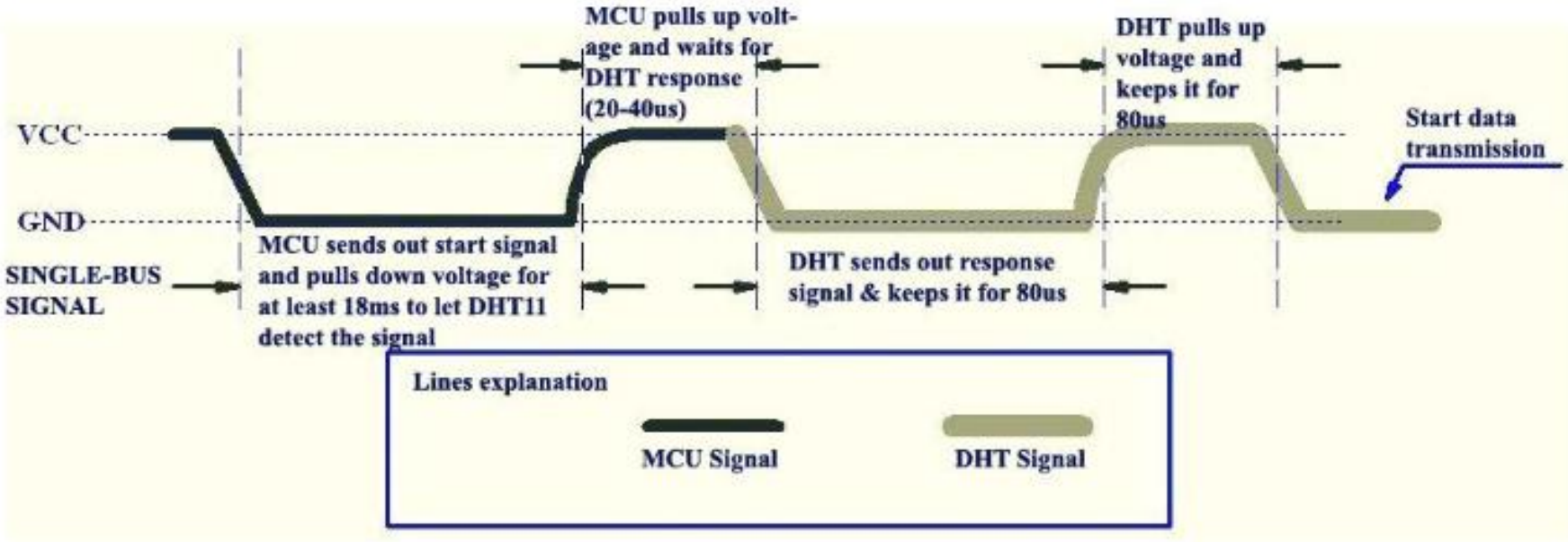


Figure 3 MCU Sends out Start Signal & DHT Responses

통신방식

5.3 DHT Responses to MCU (Figure 3, above)

Once DHT detects the start signal, it will send out a low-voltage-level response signal, which lasts 80us. Then the programme of DHT sets Data Single-bus voltage level from low to high and keeps it for 80us for DHT's preparation for sending data.

When DATA Single-Bus is at the low voltage level, this means that DHT is sending the response signal. Once DHT sent out the response signal, it pulls up voltage and keeps it for 80us and prepares for data transmission.

When DHT is sending data to MCU, every bit of data begins with the 50us low-voltage-level and the length of the following high-voltage-level signal determines whether data bit is "0" or "1" (see Figures 4 and 5 below).

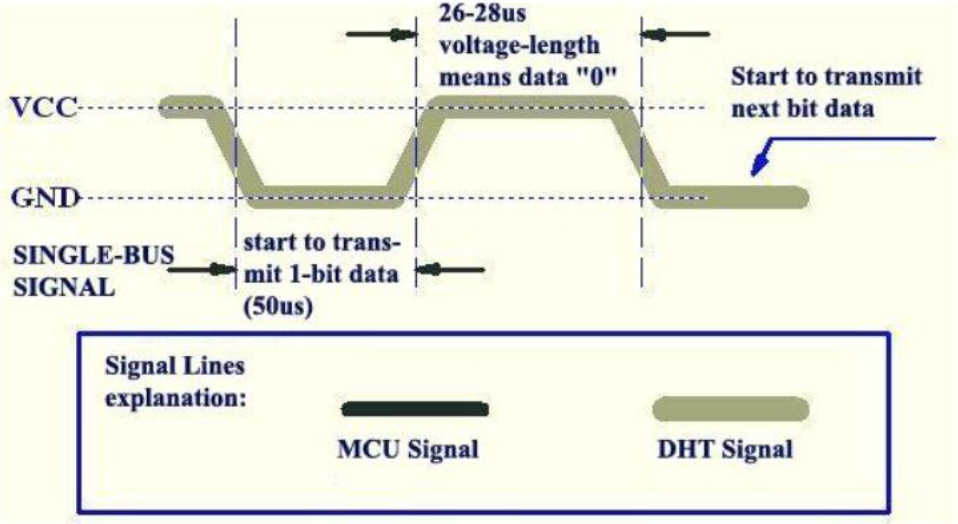


Figure 4 Data "0" Indication

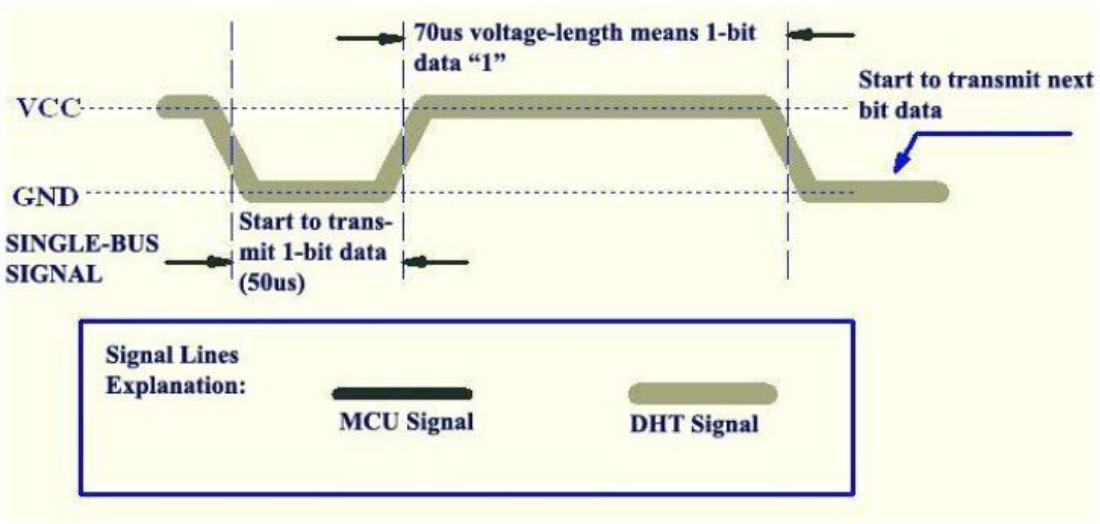
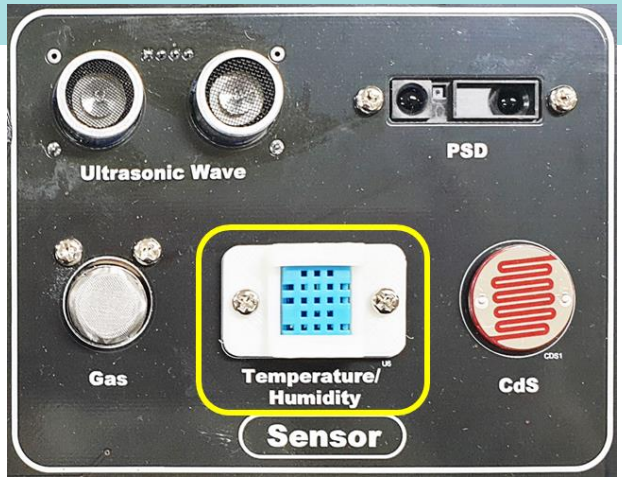
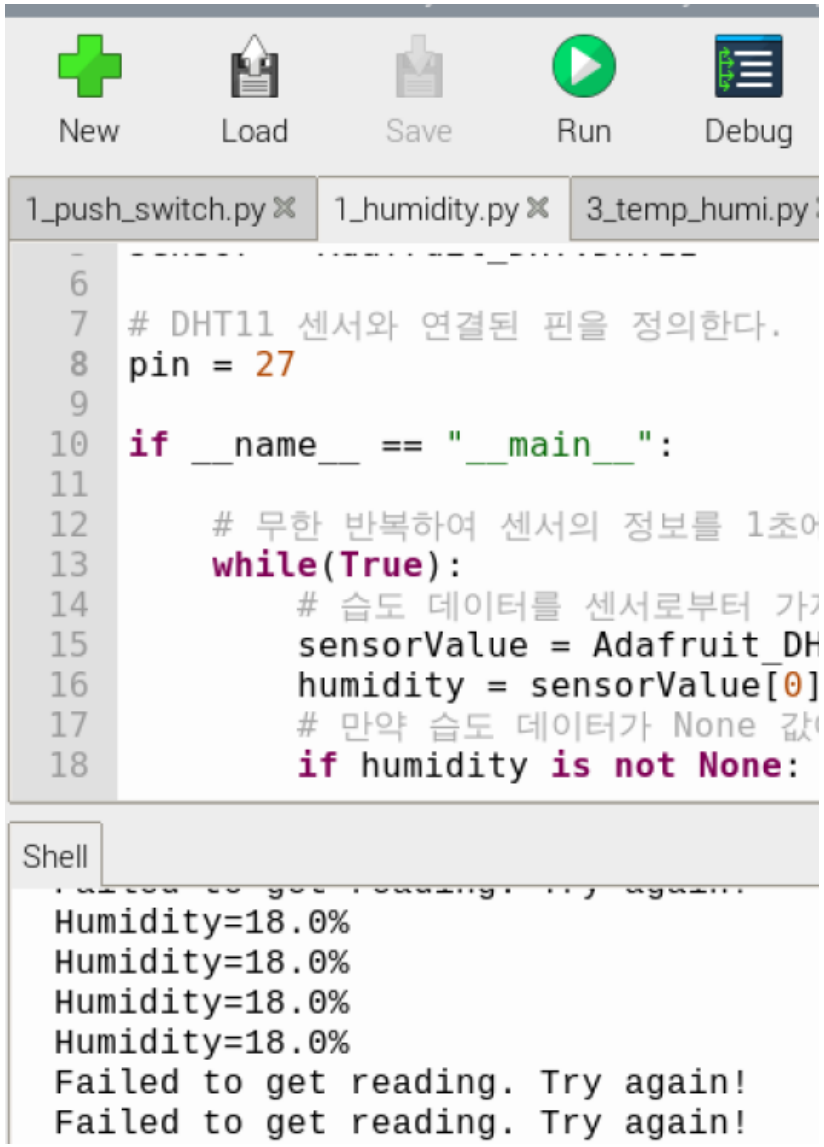


Figure 5 Data "1" Indication

습도검출




```
import Adafruit_DHT
import time

# DHT11 센서 객체를 선언한다.
sensor = Adafruit_DHT.DHT11

# DHT11 센서와 연결된 핀을 정의한다.
pin = 27

if __name__ == "__main__":

    # 무한 반복하여 센서의 정보를 1초에 한번씩 출력한다.
    while(True):
        # 습도 데이터를 센서로부터 가져온다.
        sensorValue = Adafruit_DHT.read(sensor, pin)
        humidity = sensorValue[0]
        # 만약 습도 데이터가 None 값이 아니라면 값을 출력한다.
        if humidity is not None:
            print('Humidity={}%'.format(humidity))
        else:
            print('Failed to get reading. Try again!')

        # 1초간 대기한다.
        time.sleep(1.0)
```



온도검출

```
import Adafruit_DHT
import time

# DHT11 센서 객체를 선언한다.
sensor = Adafruit_DHT.DHT11

# DHT11 센서와 연결된 핀을 정의한다.
pin = 27

if __name__ == "__main__":

    # 센서의 정보를 1초에 한번씩 출력한다.
    while(True):
        # 온도 데이터를 센서로부터 가져온다.
        sensorValue = Adafruit_DHT.read(sensor, pin)
        temperature = sensorValue[1]
        # 만약 온도 데이터가 None 값이 아니라면 값을 출력한다.
        if temperature is not None:
            print('Temperatrue={}°C'.format(temperature))
        else:
            print('Failed to get reading. Try again!')

        # 1초간 대기한다.
        time.sleep(1)
```

온/습도검출

```
import Adafruit_DHT
import time

# DHT11 센서 객체를 선언한다.
sensor = Adafruit_DHT.DHT11

# DHT11 센서와 연결된 핀을 정의한다.
pin = 27

if __name__ == "__main__":

    # 무한 반복하여 센서의 정보를 1초에 한번씩 출력한다.
    while(True):
        # 온습도 데이터를 센서로부터 가져온다.
        humidity, temperature = Adafruit_DHT.read(sensor, pin)

        # 만약 온습도 데이터가 None 값이 아니라면 값을 출력한다.
        if humidity is not None and temperature is not None:
            print('')
            print('Temp={}°C  Humidity={}%'.format(temperature, humidity))
        else:
            print('.',end='')

        # 1초간 대기한다.
        time.sleep(1.0)
```

WiringPi 온습도센서제어 -1

```
#include <wiringPi.h>
#include <stdio.h>

#define MAXTIMINGS 85
#define DHTPIN   2

int dhtVal[5] = { 0, 0, 0, 0, 0 };
void readData()
{
    int laststate = HIGH;
    int counter = 0;
    int j = 0, i;
    float f; /* fahrenheit */

    dhtVal[0] = dhtVal[1] = dhtVal[2] = dhtVal[3] = dhtVal[4] = 0;

    /* pull pin down for 18 milliseconds */
    pinMode(DHTPIN, OUTPUT);
    digitalWrite(DHTPIN, LOW);
    delay(18);

    /* then pull it up for 40 microseconds */
    digitalWrite(DHTPIN, HIGH);
    delayMicroseconds(40);

    /* prepare to read the pin */
    pinMode(DHTPIN, INPUT);
```

```
    /* detect change and read data */
    for (i = 0; i < MAXTIMINGS; i++)
    {
        counter = 0;
        while (digitalRead(DHTPIN) == laststate)
        {
            counter++;
            delayMicroseconds(1);
            if (counter == 255)
            {
                break;
            }
        }

        laststate = digitalRead(DHTPIN);

        if (counter == 255)
            break;

        /* ignore first 3 transitions */
        if ((i >= 4) && (i % 2 == 0))
        {
            /* shove each bit into the storage bytes */
            dhtVal[j / 8] <<= 1;
            if (counter > 26)
                dhtVal[j / 8] |= 1;
            j++;
        }
    }
```

WiringPi 온습도센서제어-2

```
 /*
     * check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
     * print it out if data is good
     */
    if ((j >= 40) &&
        (dhtVal[4] == ((dhtVal[0] + dhtVal[1] + dhtVal[2] + dhtVal[3]) & 0xFF)))
    {
        f = dhtVal[2] * 9. / 5. + 32;
        printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)₩n",
            dhtVal[0], dhtVal[1], dhtVal[2], dhtVal[3], f);
    }
    else {
        printf("Invalid Data! ₩n");
    }
    return;
}
```

```
int main(void)
{
    printf("Raspberry Pi wiringPi DHT11 Temperature test
programm₩n");

    if (wiringPiSetup() == -1)
        return -1;

    while (1)
    {
        readData();
        delay(3000); /* wait 3sec to refresh */
    }

    return(0);
}
```

WiringPi 온습도센서제어-2

```
bready@raspberryAI:~/AISW $ sudo nano dht11.c
```

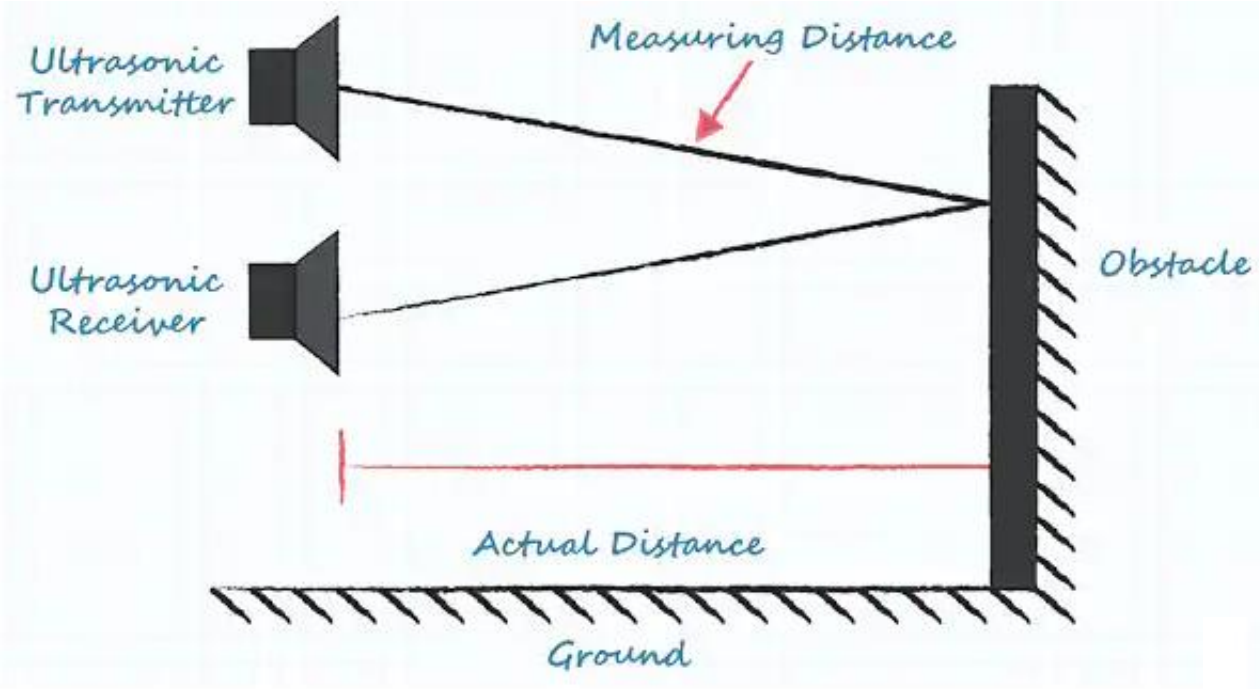
```c
/*
 *  wiringDHT11.c:
 * Simple test program to test the wiringPi functions
 * DHT11 test
 */

#include <wiringPi.h>
#include <stdio.h>



#define MAXTIMINGS 85
#define DHTPIN   2



int dhtVal[5] = { 0, 0, 0, 0, 0 };



void readData()
{
    int laststate = HIGH;
    int counter = 0;
    int j = 0, i;
    float f; /* fahrenheit */

    dhtVal[0] = dhtVal[1] = dhtVal[2] = dhtVal[3] = dhtVal[4] = 0;
```

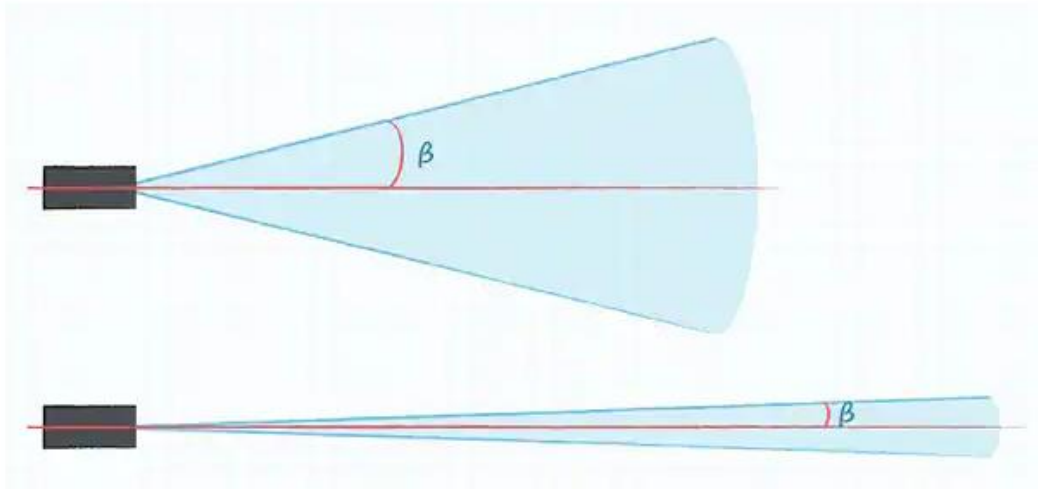
WiringPi 온습도센서제어-2

```
bready@raspberryAI:~/AISW $ gcc -o temp dht11.c -lwiringPi
```

```
bready@raspberryAI:~/AISW $ sudo ./temp
Raspberry Pi wiringPi DHT11 Temperature test program
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Humidity = 18.0 % Temperature = 26.6 *C (78.8 *F)
Humidity = 13.0 % Temperature = 26.5 *C (78.8 *F)
Humidity = 13.0 % Temperature = 26.5 *C (78.8 *F)
```

초음파 센서 제어

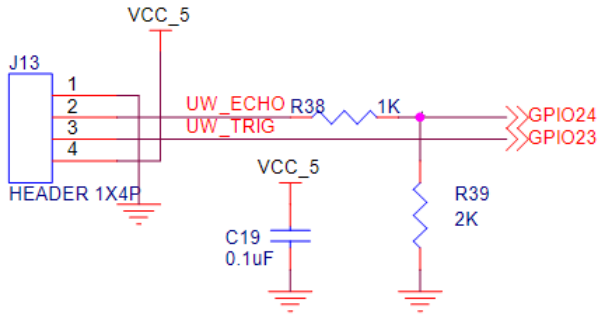




$$t = \frac{2 \times L}{V_S}$$

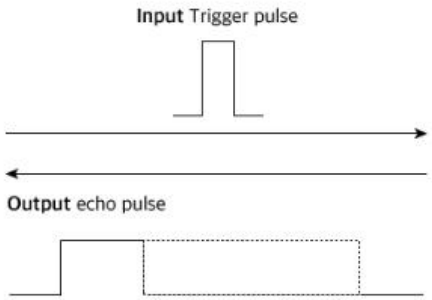
L : 물체와의 거리(m)
V_S : 음속(m/s)
t : 신호가 되돌아 올 때까지 걸리는 시간(s)

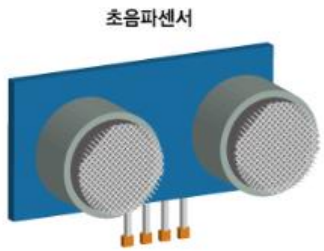
초음파 센서 제어

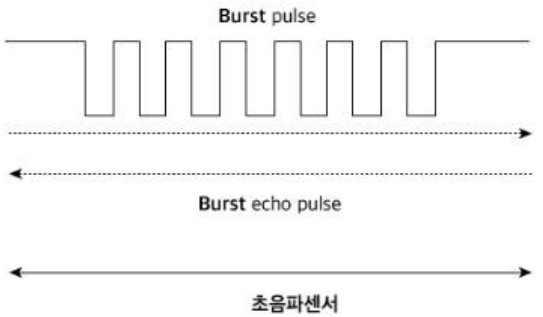



```
x : 34000 = t/2 : 1000000
x = t * 0.017 또는 x = t / 58
```

- t = 2 * 0.01 / 340 = 58.824µs
- 초음파가 1cm를 이동하는데 걸리는 시간은 약 29µs
- 초음파가 반사된 물체와의 거리:
- 거리(cm) = duration (왕복에 걸린 시간) / 29 / 2 (왕복)







초음파 센서 제어(파이썬)

Thonny - /home/bready/Desktop/Example/01.HW_Example/10_Ultrasonic/1_u

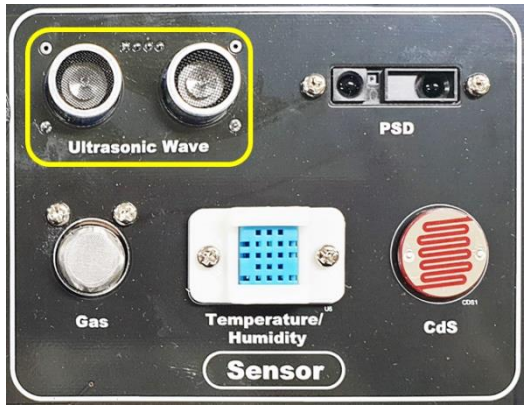
New Load Save Run Debug Over Into Out Stop

1_ultrasonic.py

```
1  import RPi.GPIO as GPIO
2  import time
3
4  # BCM 핀맵을 사용한다
5  GPIO.setmode(GPIO.BCM)
6
7  # 초음파센서에서 사용되는 핀
8  trig = 23
9  echo = 24
10
11 # trig핀을 출력으로 설정한다
12 GPIO.setup(trig, GPIO.OUT)
13
```

Shell

```
Distance : 23.71cm
Distance : 20.1cm
Distance : 22.46cm
Distance : 26.2cm
Distance : 19.21cm
Distance : 192.87cm
```



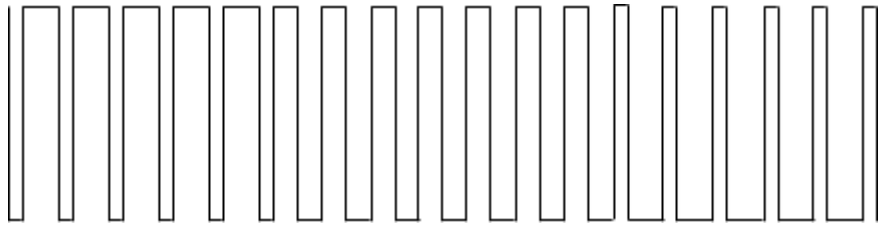
초음파 센서 제어(파이썬)

```
import RPi.GPIO as GPIO
import time

# BCM 핀맵을 사용한다
GPIO.setmode(GPIO.BCM)

# 초음파센서에서 사용되는 핀
trig = 23
echo = 24

# trig핀을 출력으로 설정한다
GPIO.setup(trig, GPIO.OUT)

# echo핀을 입력으로 설정한다
GPIO.setup(echo, GPIO.IN)

try :
    # 스레드는 반복하며 초음파 센서의 거리를 측정한다.
    while True :
        # 거리 측정을 위해 초음파를 쏜다
        GPIO.output(trig, False)
        time.sleep(0.5)
        GPIO.output(trig, True)
        time.sleep(0.00001)
        GPIO.output(trig, False)

        # 초음파 출력 후의 시간과 다시 받았을때의 시간을 계산한다
        while GPIO.input(echo) == 0 :
            signal_Start = time.time()

        while GPIO.input(echo) == 1 :
            signal_End = time.time()

        # 응답 받은 시간과 초음파의 속도 등으로 거리를 계산한다
        responseDuration = signal_End - signal_Start
        # (34000cm/s) 왕복이므로 2를 나누어준다
        distance = responseDuration * 34000 / 2

        # 소수점 둘 째 자리까지만 출력한다
        distance = round(distance, 2)

        # 1000cm 를 초과할시 값을 표기하지 않는다
        if distance < 1000 :
            print("Distance : " + str(distance) + "cm")
        else:
            pass

# 종료 등의 키보드 인터럽트 발생시 처리 동작
except KeyboardInterrupt:
    # GPIO를 초기화한다
    GPIO.cleanup()
```

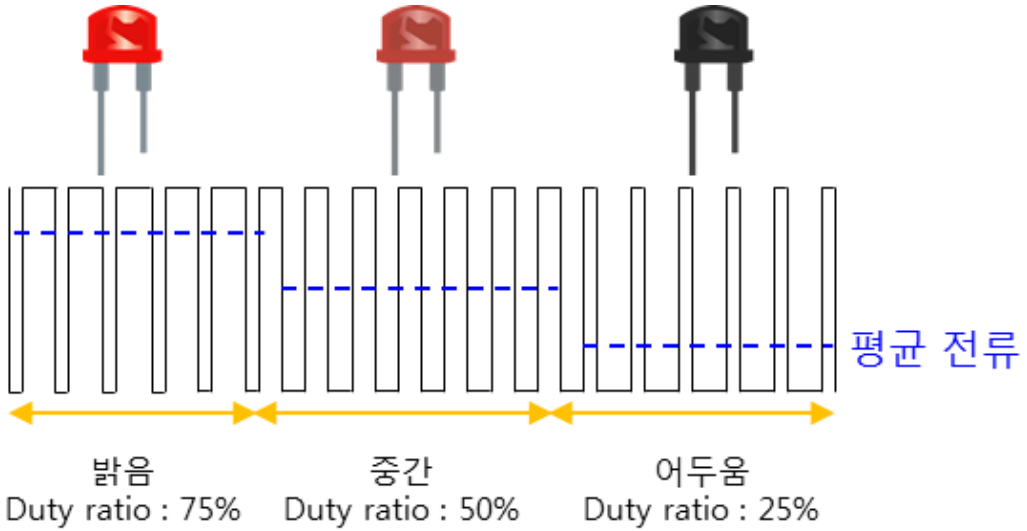
초음파 센서 제어(wiringPi)

```c
#include<stdio.h>
#include<unistd.h>
#include<time.h>
#include<wiringPi.h>

#define trigPin 4//gpio 23
#define echoPin 5//gpio 24

int main(void)
{
	int distance = 0;
	int pulse = 0;

	long startTime;
	long travelTime;

	if (wiringPiSetup() == -1)
	{
		printf("Unable GPIO Setup");
		return 1;
	}

	pinMode(trigPin, OUTPUT);
	pinMode(echoPin, INPUT);

	for (;;)
	{
		digitalWrite(trigPin, LOW);
		usleep(2);
		digitalWrite(trigPin, HIGH);
		usleep(20);
		digitalWrite(trigPin, LOW);

		while (digitalRead(echoPin) == LOW);

		startTime = micros();

		while (digitalRead(echoPin) == HIGH);
		travelTime = micros() - startTime;

		int distance = travelTime / 58;

		printf("Distance: %dcm₩n", distance);
		delay(200);
	}
}
```

피에조 부저제어(PWM제어- 파이썬)



PWM 사각파 폭



Using PWM in RPi.GPIO

To create a PWM instance:

```
p = GPIO.PWM(channel, frequency)
```

To start PWM:

```
p.start(dc)   # where dc is the duty cycle (0.0 <= dc <= 100.0)
```

To change the frequency:

```
p.ChangeFrequency(freq)   # where freq is the new frequency in Hz
```

To change the duty cycle:

```
p.ChangeDutyCycle(dc)   # where 0.0 <= dc <= 100.0
```

To stop PWM:

```
p.stop()
```

피에조 부저제어(PWM제어- 파이썬)

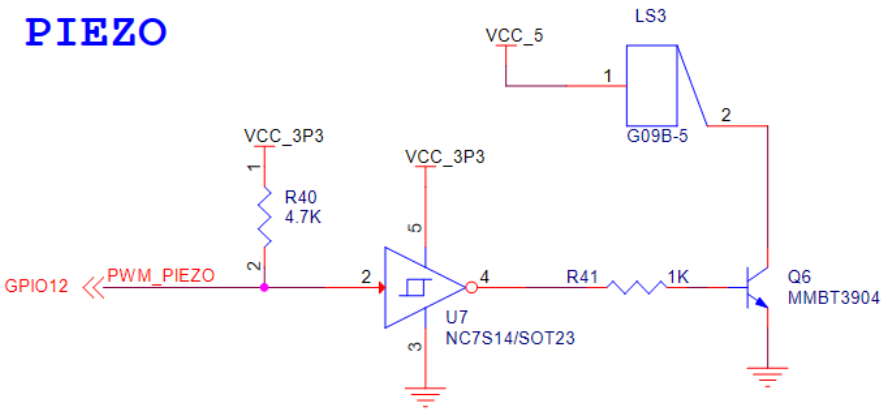
Active Buzzer

Passive Buzzer

옥타브 및 음계별 표준 주파

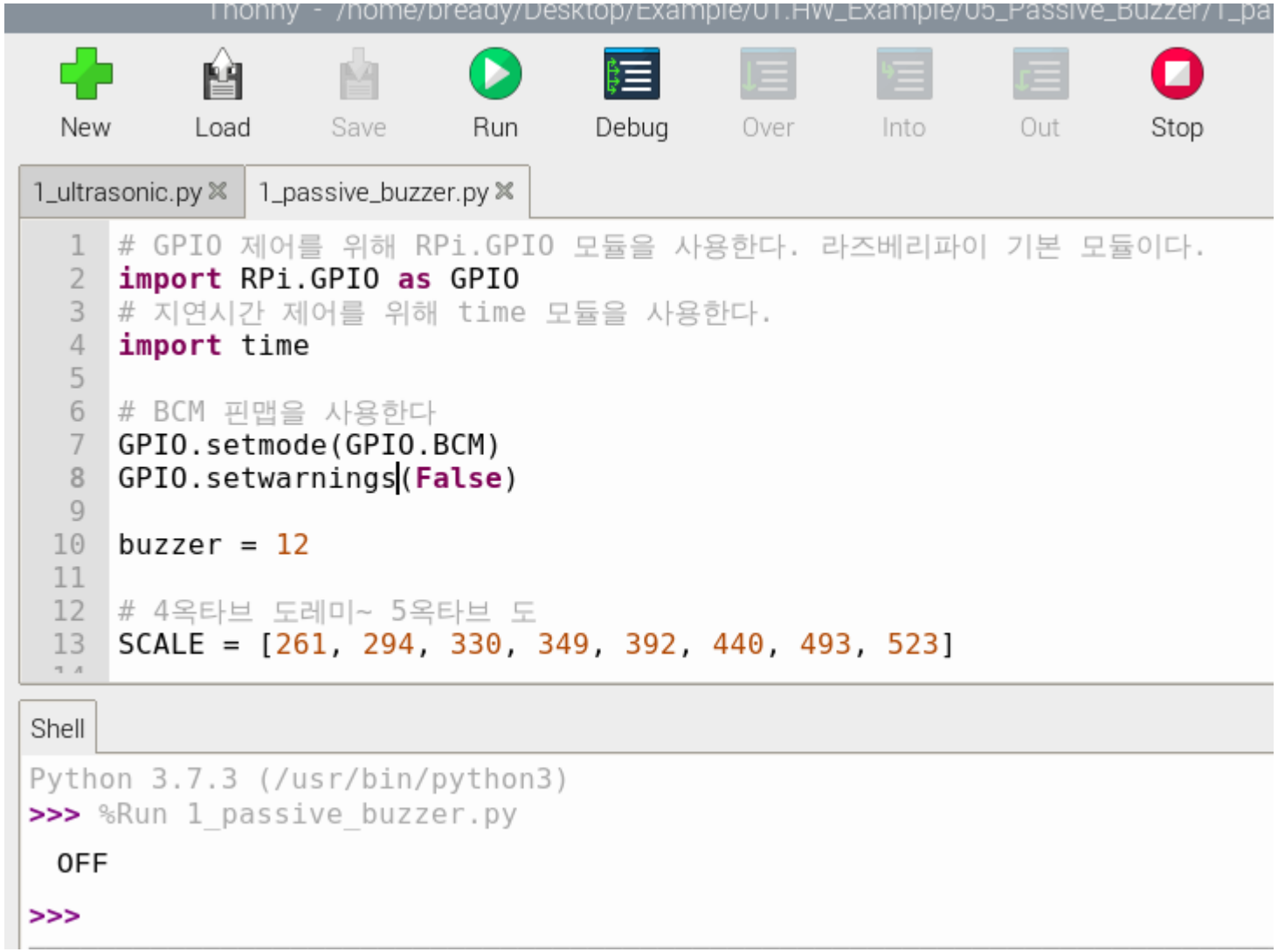
음계 \ 옥타브	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.602	2093.005	4186.009
C#	34.6478	68.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.668	587.3295	1174.659	2349.318	4698.646
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F	43.6535	87.3071	174.6141	349.2283	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9942	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	330.0000	880.0000	1760..000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7040.000
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

PIEZO



피에조 부저제어(파이썬)

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

buzzer = 12

# 4옥타브 도레미~ 5옥타브 도
SCALE = [261, 294, 330, 349, 392, 440, 493, 523]

GPIO.setup(buzzer, GPIO.OUT)
pwm = GPIO.PWM(buzzer, 100) # 100Hz

try:
    # pwm 시작
    pwm.start(100)
    # dutycycle 50%
    pwm.ChangeDutyCycle(50)

    for i in SCALE:
      pwm.ChangeFrequency(i) #주파수 변경
      time.sleep(0.5)

    # Buzzer를 끈다
    pwm.ChangeDutyCycle(100)
    print("OFF")

# 종료 등의 키보드 인터럽트 발생시 처리 동작
except KeyboardInterrupt:
        # Buzzer를 끈다
        GPIO.output(buzzer,GPIO.LOW)
        # GPIO를 초기화한다
        GPIO.cleanup()
```



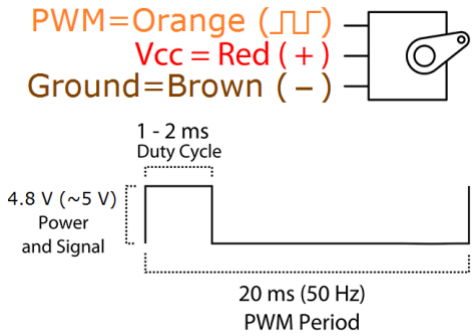
피에조 부저제어(wiringPi)

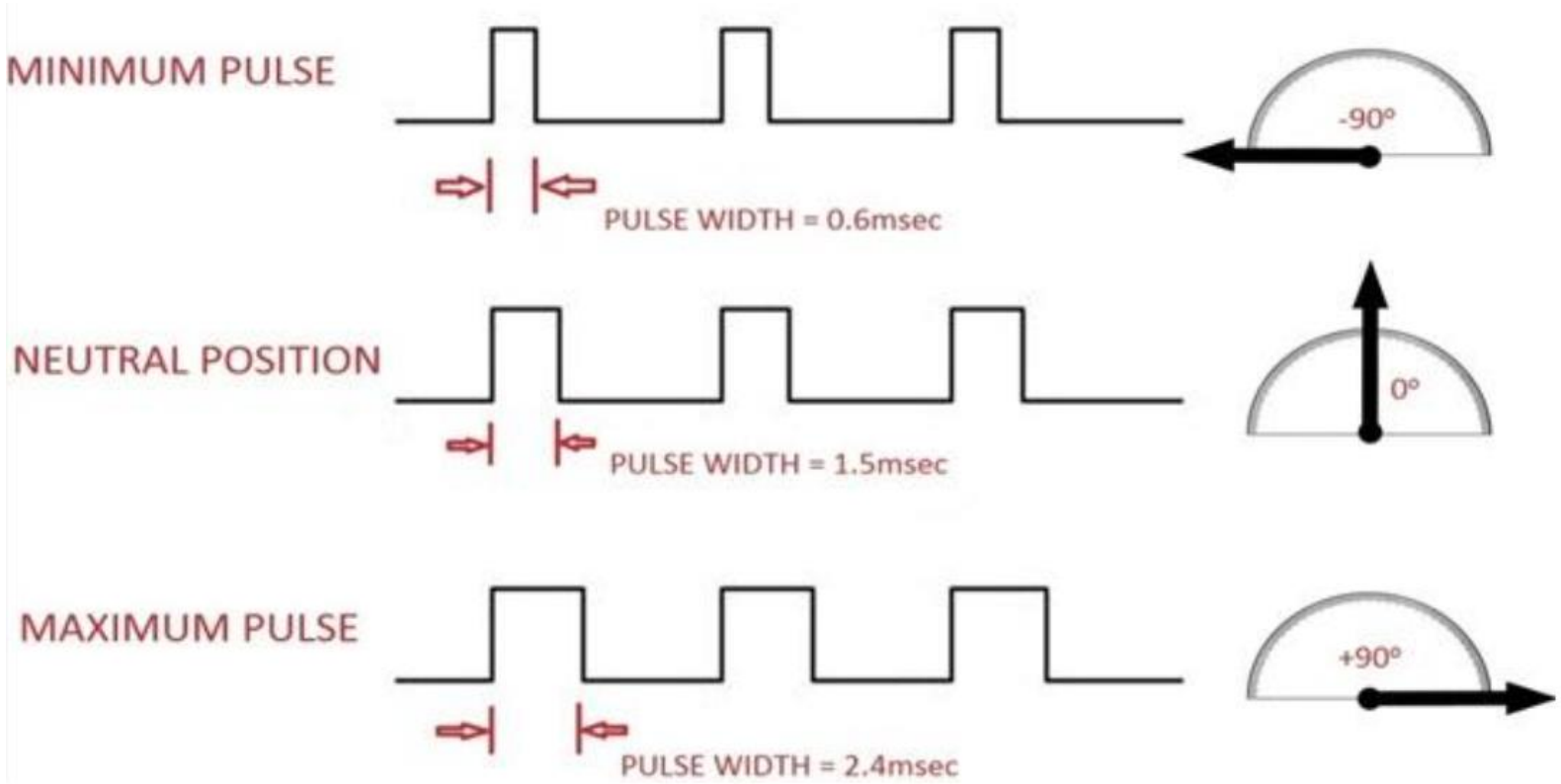
```c
#include <wiringPi.h>
#include <softTone.h>
#include <stdio.h>
#define BuzzPin 26
int song_1[] = { 261, 294, 330, 349, 392, 440, 493, 523 };

int main(void)
{
    int i, j;
    if (wiringPiSetup() == -1) {
        printf( " WiringPi initialization failed !");
        return 1;
    }
    if (softToneCreate(BuzzPin) == -1) {
        printf("Soft Tone Failed !");
        return 1;
    }
    printf("Sound is generated...₩n");
    for (i = 0; i < sizeof(song_1) / 4; i++) {
        softToneWrite(BuzzPin, song_1[i]);
        delay( 500);
    }
    return 0;
}
```

```
bready@raspberryAI:~/AISW $ gcc -o buzzer2 buzzer2.c -lwiringPi
bready@raspberryAI:~/AISW $ ./buzzer2
```

Servo 모터제어

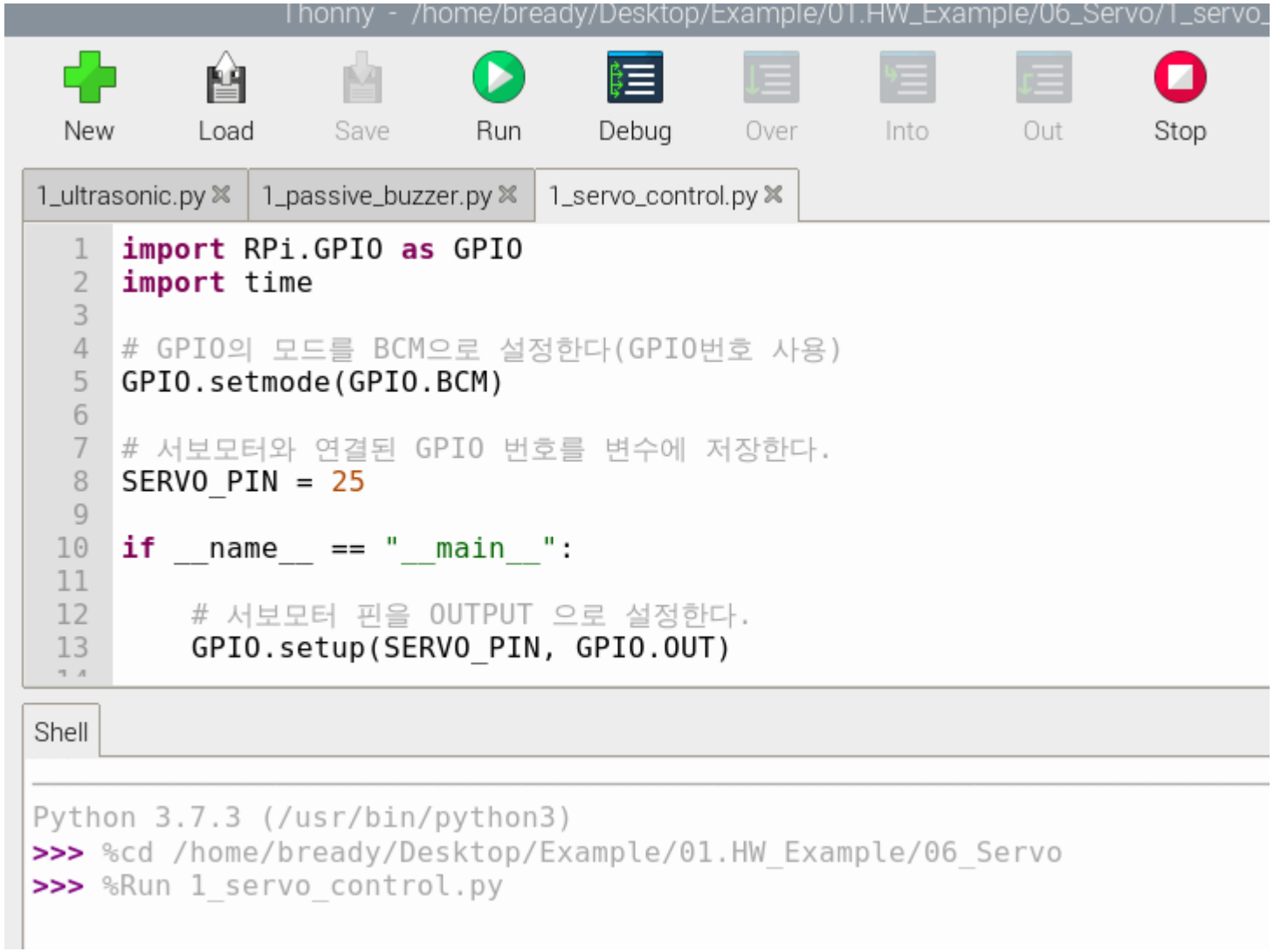

TOWER PRO
MICRO SERVO


PWM=Orange (⎍⎍)
Vcc = Red (+)
Ground=Brown (–)
1 - 2 ms
Duty Cycle
4.8 V (~5 V)
Power
and Signal
20 ms (50 Hz)
PWM Period


MINIMUM PULSE
-90°
PULSE WIDTH = 0.6msec
NEUTRAL POSITION
0°
PULSE WIDTH = 1.5msec
MAXIMUM PULSE
+90°
PULSE WIDTH = 2.4msec

Servo 모터제어

```
import RPi.GPIO as GPIO
import time

# GPIO의 모드를 BCM으로 설정한다(GPIO번호 사용)
GPIO.setmode(GPIO.BCM)

# 서보모터와 연결된 GPIO 번호를 변수에 저장한다.
SERVO_PIN = 25

if __name__ == "__main__":

    # 서보모터 핀을 OUTPUT 으로 설정한다.
    GPIO.setup(SERVO_PIN, GPIO.OUT)

    # 서보모터를 제어할 pwm 모듈을 저장한다.
    servo_pwm = GPIO.PWM(SERVO_PIN, 50)

    # 서보모터 핀의 출력을 초기화한다.
    servo_pwm.start(0)

    # 서보모터를 반복하여 0 -> 180 -> 90 순서로 동작시킨다.
    try:

        while(True):
            # 서보모터의 각도를 0도로 제어한다.
            servo_pwm.ChangeDutyCycle(7.5)
            time.sleep(1)

            # 서보모터의 각도를 -90도로 제어한다.
            servo_pwm.ChangeDutyCycle(2.5)
            time.sleep(1)

            # 서보모터의 각도를 0도로 제어한다.
            servo_pwm.ChangeDutyCycle(7.5)
            time.sleep(1)

            # 서보모터의 각도를 90도로 제어한다.
            servo_pwm.ChangeDutyCycle(12.5)
            time.sleep(1)

    # 키보드 인터럽트, 에러 등으로 소스가 종료될 경우 GPIO를 초기화한 후 종료한다.
    finally:
        GPIO.cleanup()
```



Servo 모터제어

```c
#include <wiringPi.h>
#include <stdio.h>
#include <softPwm.h>
#define PIN 6
int main()
{
    if (wiringPiSetup() == -1)
        return 1;
    softPwmCreate(PIN, 0, 50);
    softPwmWrite(PIN, 7); //
    delay(600); //600ms 동안 softPwmWrite()상태가 지속됨
    softPwmWrite(PIN, 2); //
    delay(600);
    softPwmWrite(PIN, 7); //
    delay(600); //600ms 동안 softPwmWrite()상태가 지속됨
    softPwmWrite(PIN, 12); //
    delay(600); //600ms 동안 softPwmWrite()상태가 지속됨
    return 0;
}
```

```
bready@raspberryAI:~/AISW $ gcc -o servomotor servomotor.c -lwiringPi
bready@raspberryAI:~/AISW $ ./servomotor
```